Navigation Planning in Probabilistic Roadmaps with Uncertainty

Michael Kneebone and **Richard Dearden**
School of Computer Science
University of Birmingham
Birmingham, B15 2TT, UK
mlk,rwd@cs.bham.ac.uk

Abstract

Probabilistic Roadmaps (PRM) are a commonly used class of algorithms for robot navigation tasks where obstacles are present in the environment. We examine the situation where the obstacle positions are not precisely known. A subset of the edges in the PRM graph may possibly intersect the obstacles, and as the robot traverses the graph it can make noisy observations of these *uncertain edges* to determine if it can traverse them or not. The problem is to traverse the graph from an initial vertex to a goal without taking a blocked edge, and to do this optimally the robot needs to consider the observations it can make as well as the structure of the graph. In this paper we show how this problem can be represented as a POMDP. We show that while too large to be solved with exact methods, approximate point based methods can provide a good quality solution. While feasible for smaller examples, this approach isn't scalable. By exploiting the structure in the belief space, we can construct an approximate belief-space MDP that can be solved efficiently. We demonstrate that this gives near optimal results in most cases while achieving an order of magnitude speed-up in policy generation time.

Introduction

This paper examines the problem of path planning in the face of obstacles whose positions are not known with certainty. The approach we take is to extend Probabilistic Roadmap (PRM) planning to handle uncertainty and observations.

Probabilistic Roadmaps (Kavraki and Latombe 1998) are a popular technique for path planning in high dimensional spaces. They are applicable in many situations including robot arm motion planning and path planning for mobile agents. PRMs generate a random graph with vertices representing reachable robot poses and edges representing motion from one pose to another. This graph is then searched for a path from the initial state to the goal. Their success is due to the fact that they reduce search in a large continuous space into a graph search. However, most PRM approaches rely on the assumption that the planner knows the locations of all obstacles in the environment prior to building the roadmap graph or plotting a route. There are many instances where this is not the case, for example when moving over longer distances than the sensors can measure, or when obstacles are obscured by others in front of them.

Our approach assumes the agent is initially uncertain about obstacle locations and hence about which graph edges are blocked. The system builds a plan to reach the goal that makes observations of these *uncertain edges* to determine which are blocked and which it can use. The planned route is then based on both the agent's belief about which edges are usable and on the value of being able to make better observations about the obstacle locations. This contrasts with previous approaches because we explicitly reason about the information we receive while executing the plan.

We will assume that most edges in the graph can be traversed with certainty, and that there are only a relatively small number of uncertain edges. We can think of this as a decision-theoretic model identification problem where there are m uncertain edges, hence 2^m possible models. By representing this as a POMDP and then applying a number of techniques to exploit model structure, we show that the problem can be made tractable without significantly compromising plan quality. We provide a variety of approximations that trade off solution time and computation time.

Probabilistic Roadmaps

Probabilistic Roadmap (Kavraki and Latombe 1998) planners find paths for robotic agents by searching over a graph of possible configurations of that agent. By sampling random values for each degree of freedom of the agent to create a complete configuration (pose), a PRM planner is capable of "exploring" the entire configuration space (C-space) for a given agent. Not all configurations in the space will be usable due to obstacles or self-collisions, so the n-dimensional space is conceptually divided into two non-contiguous areas named C_{free} and C_{obst} for free and obstructed configurations respectively. A PRM graph is built in the preprocessing phase of the planner by using a sampling algorithm which randomly generates configurations for the agent and then either accepts or rejects the generated poses. The simplest strategy is to reject all poses that intersect C_{obst}.

The graph is completed by taking each sampled vertex and attempting to create k edges to neighbouring vertices by checking for a collision-free route between the two vertices. In the query phase of PRM, a completed graph can then be searched rapidly for paths between two arbitrary poses using standard graph search algorithms.

To model uncertain obstacles we assume we have proba-

bility distributions over the positions of the obstacle vertices. We generate a PRM graph in advance in which some edges may intersect the uncertain obstacles. We then build a path plan that traverses the graph optimally, given the fact that observations will be made as we get closer to the obstacles that will allow us to determine which edges are obstacle-free. In real world domains we can determine edge dependencies and observation probabilities by sampling obstacle positions from their distributions.

Model Formulation

To formulate this uncertain PRM in a tractable way, we represent the obstacle locations only in terms of their effect on the uncertain edges, and consider the problem as one of efficiently traversing the PRM graph.

Let $G = \{V, E\}$ be a graph where the vertices $V = \{v_1, ..., v_n\}$ represent robot poses, and the edges $E = \{e_1, ...e_k\}$, where $e_i = \langle v, v' \rangle$, represent paths between poses (we assume that if there is a path $\langle v, v' \rangle$, then there is a corresponding path $\langle v', v \rangle$). We also identify locations v_S and v_G, the start and goal poses. Each edge e has a cost c_e of traversing it.

When we visit a vertex of the graph, we make observations of all the uncertain edges from our new location, and we receive information about whether the edges are obstacle-free[1]. We will assume that for each edge e_i we observe either f_i, meaning that we observe the edge to be collision-free, or b_i if the edge appears blocked. These observations are uncertain, in that if the edge is collision-free, we may still observe b_i some of the time, and vice versa. Assume there are m uncertain edges. We write $\overline{o} =< o_1, \ldots, o_m >$ for an observation of each of the uncertain edges, where o_i is either f_i or b_i.

Although we can't observe it directly, there is in fact a true state of the world in which each edge is either collision-free or blocked. Let $W = \{w_1, \ldots, w_{2^m}\}$ be the set of all such worlds. We write $P(\overline{o}|v_i, w_j)$ for the probability of making observation $\overline{o}$ from location v_i if the true state of the world is w_j. While we've abstracted away the obstacle locations in this representation, the fact that the traversability of the edges is based on obstacle locations is important because it implies that for close edges, the probability that the edges are blocked may not be independent. If the edges are independent, then the likelihood that world w is the true state of the world is simply the product of the likelihoods of each edge being collision-free or blocked. If this is not true (for example, where one obstacle is likely to intersect two edges), then the edge probabilities are dependent, and the probability of each world must be maintained separately in the belief state.

For the purpose of exposition, to find an optimal path in a PRM graph with no uncertain edges we could represent the problem as a Markov decision problem (MDP) where the state space $S = V$, the action space $A = \{\text{goto-}v : v \in V\}$ (note that not all actions in this model are applicable in every state), and the reward and transition functions are given by:

$$R(s, \text{goto-}v) = \begin{cases} -c_e & \text{if } e = \langle s, v \rangle \text{ and } s \neq v_G \\ -\infty & \text{otherwise} \end{cases}$$

where v_G is a goal vertex, and

$$P(s_i, \text{goto-}s_j, s_j) = \begin{cases} 1 & \text{if } \langle s_i, s_j \rangle \in E \\ 0 & \text{otherwise} \end{cases}$$

Intuitively, the vertices correspond to states, edges correspond to actions, all the transition probabilities are 0 except for those that correspond to edges in the graph, and rewards correspond to costs of traversing edges. This is a shortest cost path problem with the goal being an absorbing state.

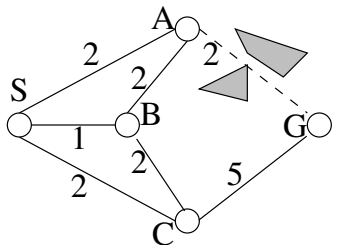

Figure 1: A small example PRM graph. This example graph has one uncertain edge, which initially has $P(\text{blocked}) = 0.5$ (the dotted line indicates the uncertain edge). The state of the edge can be observed with certainty from vertices A and B.

With uncertain graph edges present, the optimal route may depend on observations made along it and the shortest path may no longer be the most efficient. Figure 1 illustrates this problem on a small PRM with only one uncertain edge. For this example we assume that vertices A and B are close enough to the obstacle to observe with certainty if edge AG is blocked. If the edge is collision-free, the optimal path is to move from S to A to G. If the edge is blocked, then S to C to G is optimal, but if our current belief is that the edge is blocked with $p = 0.5$, the optimal behaviour is to move to B, observe the edge and then move to A or C as appropriate. The optimal policy under uncertainty is different from the optimal policies in either of the possible worlds.

As the agent always knows which vertex is the current location, the choice of next vertex depends solely on the current state, thus the model is Markovian. Each of the 2^m worlds is then a possible MDP, but we are uncertain which MDP is the correct one, so we must take information gathering actions as we traverse the graph by making observations.

POMDP Representation

We represent the model identification problem as a partially observable MDP. A POMDP is a tuple $< S, A, O, T, H, R >$ where S and A are as before, O is the set of possible observations, $T = P(s, a, s')$ is the transition function that governs how an action changes the state of the system, $H = P(o|s, a, s')$ is the observation function that governs how likely each observation is given a state, action and resulting state, and $R = r(s, a)$ is the

[1] If not all edges are visible, we assume we receive an observation that gives no information about those edges.

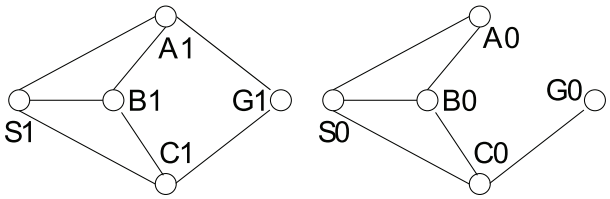

Figure 2: The POMDP model of the graph in Figure 1. The states and actions are shown, the rewards are as before. At each vertex, we observe the vertex (letter) we are at. We additionally observe A→G as free at $A1$ and $B1$ and as blocked at $A0$ and $B0$.

reward function which specifies the immediate reward (or cost) of doing a particular action in a state.

To translate our model identification problem into a POMDP, the state space is the union of the state spaces for the MDPs for each possible model, i.e. the set of all the states in each world (giving $n2^m$ states). For Figure 1 the ten ($m = 1$ and $n = 5$) possible states are shown in Figure 2. The action set is unchanged, consisting of one action for each PRM vertex: five in this case. The rewards and transition functions are the unions of the set of MDPs.

We define the function valid(v, v', w) to represent the fact that there is a collision-free edge in world w between v and v'. Formally, valid(v, v', w) is true if $\exists e \in E : e = \langle v, v' \rangle$ and e is collision-free in w.

We can now formally define the model identification POMDP as follows:

- $S = V \times W$ as described above. Each state is a combined $\langle v \in V, w \in W \rangle$ pair.
- $A = \{\text{goto-}v : v \in V\}$ as above.
- $O = \mathcal{P}(\overline{o})$ an observation of all the uncertain edges, $\overline{o} = \langle o_1, o_2, \ldots, o_m \rangle$ where o_i is an observation of uncertain edge i.
- T is defined as follows:

$$P(\langle v, w \rangle, \text{goto-}v', \langle v'', w' \rangle) = \begin{cases} 1 & \text{if } w = w',\ v' = v'', \text{ and valid}(v, v', w) \\ 0 & \text{otherwise} \end{cases}$$

  T defines the probability of a transition from one state to another. All transitions have probability zero except transitions where the PRM states being moved between are in the same world, and there is a collision-free edge in that world between those two states.

- H is defined as follows:

$$P(\overline{o} | \langle v, w \rangle, \text{goto-}v', \langle v', w \rangle) = \begin{cases} P(\overline{o} | v', w) & \text{if valid}(v, v', w) \\ 0 & \text{otherwise} \end{cases}$$

  For brevity we have relaxed the notation compared to the definition of T. H is the probability of an observation given a state, action, and resulting state, and in this case is defined solely by the resulting state. Whenever we perform an action in the POMDP, we get an observation of whether each uncertain edge is collision-free or blocked.

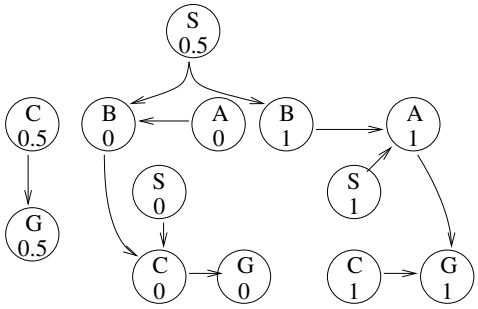

Figure 3: The optimal policy for the example graph in Figure 1. The policy shown is for starting from S with a probability of 0.5 that the uncertain edge is blocked. The split arrow from S indicates that the resulting state is unknown.

- R is defined as in the MDP model above by:

$$R(\langle v, w \rangle, \text{goto-}v') = \begin{cases} -c_e & \text{if } e = \langle v, v' \rangle, \text{ valid}(v, v', w) \\ & \text{and } v \neq v_G \\ -\infty & \text{otherwise} \end{cases}$$

The belief state is a distribution only over the possible worlds. If the probabilities that two edges are blocked are not independent, this is represented in the POMDP model by encoding the dependency in the belief state. The choice of initial belief state is therefore crucial—it specifies any edge dependencies to the agent in the world. We can represent the fact that two edges are always in the same state (collision-free or blocked) by making all worlds in which one is blocked and the other is not have prior probability zero. The agent is then forced to believe those worlds are impossible. In our experiments we use the initial belief states to set edges as being dependent or not as appropriate.

We can now solve this POMDP to find an optimal policy under our uncertainty about which world we are in. States reachable under any policy have non-zero probability for the subset of the $n2^m$ POMDP states which represent the current vertex, and zero for all others. Figure 3 shows the optimal policy for Figure 1. Each circle contains the PRM graph vertex and the belief that edge AG is collision-free.

A Continuous MDP Model

Solving even very small POMDPs is computationally hard. In our case, there is a lot of structure in the POMDP belief space. There are a number of general POMDP solvers that exploit structure, for example point-based solvers (Pineau, Gordon, and Thrun 2003; Spaan and Vlassis 2005), or belief compression (Roy, Gordon, and Thrun 2005). General algorithms don't specialise towards one problem domain, so we should be able to do significantly better by using algorithms specific to the type of structure present.

In the case of our domain, the only belief states that are possible are ones that are certain about which graph vertex they are in, but have uncertainty over the which of the 2^m possible worlds is the true one. We now develop an approximate MDP model that explicitly represents this structure.

Consider a problem such as the one in Figures 1 and 2 in which there is only a single uncertain edge. In this case, every possible belief state consists of an underlying MDP state plus a single probability of being in the obstacle-free world. Thus we can solve this problem using an MDP solver for continuous states (or approximately by discretising the continuous variable). When we generalise this approach to higher numbers of uncertain edges, the m uncertain edges produce 2^m possible worlds, since the edge probabilities could be dependent on one another, leading to an MDP with a state space containing $2^m - 1$ continuous variables.

To formulate this model as a discrete MDP, we discretise the belief space over each continuous variable (the probability of each model). Since zero and one are important to distinguish in these problems as they lead to smaller branching factors in the MDP, we discretise into $d + 1$ values as follows: $D = \{0, 1/d, 2/d, \ldots, (d-1)/d, 1\}$ where every value in the range $[i/d - 1/(2d), i/d + 1/(2d))$ is discretised to i/d, and 0 and 1 correspond to the ranges $[0, 1/(2d))$ and $[1 - 1/(2d), 1]$ respectively. Given such a discretisation, and a set of 2^m possible worlds, let $\widetilde{P}(w_i)$ be the probability of world i, discretised according to D. Let $\widetilde{P}_W = \{\widetilde{P}(w_1), \ldots, \widetilde{P}(w_{2^m})\}$ such that $\sum_i \widetilde{P}(w_i) = 1$ be a discretised assignment of probability to every possible world. Let $\widetilde{P} = \mathcal{P}(\widetilde{P}_W)$ be the set of all such assignments.

We can now define the discretised MDP formulation of the problem as follows:

- $S = V \times \widetilde{P}$ is the set of states. It is the cross product of the set of vertices in the PRM graph and the set of possible discretised beliefs about which world is the true one.
- $A = \{\text{goto-}v : v \in V\}$ as before.
- R, the reward function, is defined as follows:

$$R(\langle v, \widetilde{P}_W \rangle, \text{goto-}v') = \sum_w \widetilde{P}(w) R(\langle v, w \rangle, \text{goto-}v') \quad (1)$$

  where $R(\langle v, w \rangle, \text{goto-}v')$ is defined as in the POMDP formulation above.

- T, the transition function, is somewhat complex to define as we have to specify both the transition probabilities and the states that result. When we make a transition in the POMDP formulation the actual transition is deterministic in the underlying states, but the observation (which is stochastic) moves us from one belief state to another. In the MDP formulation, the transitions with non-zero probability correspond to all the observations that could be made and the resulting states are the discretised belief states that would result from making each observation. Let the transition probability be defined as follows:

$$p = P(\langle v, \widetilde{P}_W \rangle, \text{goto-}v', \langle v', \widetilde{P}'_W \rangle)$$

  We first consider the cases where p is non-zero. For this to be true, $\langle v, v' \rangle \in E$. Suppose we make an observation $\overline{o}$, then the belief state we move to is:

$$\widetilde{P}'_W = \{P_{v'}(w_1|\overline{o}, \widetilde{P}_W), \ldots, P_{v'}(w_{2^m}|\overline{o}, \widetilde{P}_W)\}$$

  where $P_{v'}(w_i|\overline{o}, \widetilde{P}_W)$ is the new probability of being in model i after observing $\overline{o}$, which is:

$$\begin{aligned} P_{v'}(w_i|\overline{o}, \widetilde{P}_W) &= \frac{P(\overline{o}|v', w_i)P(w_i)}{P(\overline{o}|v', \widetilde{P}_W)} \\ &= \frac{P(\overline{o}|v', w_i)P(w_i)}{\sum_j P(\overline{o}|v', w_j)P(w_j)} \end{aligned} \quad (2)$$

  Where $P(\overline{o}|v, w)$ is defined in the PRM graph (it is the probability of seeing observation $\overline{o}$ from vertex v if the true world is w), and $P(w)$ given by the prior world probabilities from $\widetilde{P}_W$. We now discretise $P_{v'}(w_i|\overline{o}, \widetilde{P}_W)$ to produce the new belief state after the action.

  It only remains to compute p, the probability of reaching these belief states, and this is given by:

$$\begin{aligned} p &= P(\overline{o}|v', \widetilde{P}_W) \\ &= \sum_j \widetilde{P}(w_j) P(\overline{o}|v', w_j) \end{aligned} \quad (3)$$

  For all states other than those identified above, $p = 0$.

As before, if two uncertain edges are correlated, this is reflected in the belief state over the possible worlds, so affects only the initial state in the MDP.

Having defined the MDP, we can now solve it to get an optimal (modulo the discretisation) policy for the POMDP, and hence for the original PRM problem. We will refer to this MDP formulation as the *dependent MDP* as it allows us to represent belief states where there are arbitrary dependencies between the probabilities that edges are collision-free. The dependent MDP formulation is significantly quicker to solve than the POMDP formulation of the problem.

LAO*

The starting state specifies the agent's prior belief over which world is the correct one, yet standard MDP planners compute a policy for all states. Particularly in this domain, many MDP states are never visited under any sensible policy. An obvious improvement therefore is to only consider the value of states reachable from the start state.

Many modern methods take this approach in both MDP (such as RTDP (Barto, Bradtke, and Singh 1993), envelope methods (Dean et al. 1995)) and POMDP planning (Ross et al. 2008). Since the MDPs we are solving are stochastic shortest path problems, LAO* (Hansen and Zilberstein 2001) is a natural choice of solution algorithm. LAO* is a heuristic search algorithm which builds a graph forwards from the start state and expands leaf nodes according to a heuristic akin to how A* explores a graph. Value iteration updates (backups) are applied to those states in the current best solution graph (BSG). LAO* terminates when there are no remaining unexpanded nodes in the BSG and all node values in it have converged values under value iteration.

Our problem domain allows admissible heuristics to be easily devised for the algorithm. We use the shortest-path cost to the goal assuming all edges are free as the heuristic. This is efficiently computed using Dijkstra's algorithm on the PRM graph. LAO* exhibits large gains in performance compared to a standard MDP solver that uses value iteration over the full state space.

For small problems we can compare value iteration to LAO* directly. On a graph of 6 vertices with 3 uncertain edges and a discretisation of $d=0.1$ the resulting MDP contains 7,986 states. The MDP solver took 205.8 seconds to generate the optimal policy while LAO* produced a solution in 1.5 seconds from a start state where all three uncertain edges were independently blocked with probability 0.5. The BSG contained just 376 nodes. To evaluate the policies LAO* finds, we implemented a simulator which randomly samples 'true' world states according to the initial belief distribution and executes the policy in each one. The total incurred cost is recorded when the goal is reached. Trials terminate with failure if a preset maximum number of steps is exceeded or the policy tries to traverse a blocked edge.

Approximate LAO*

A disadvantage for LAO* in this domain is the large number of MDP states (nodes) it has to expand and evaluate due to the branching factor. Even small graphs can lead to large numbers of reachable belief states and all nodes that may be optimal must be evaluated. When LAO* expands a leaf and an observation is made, the successor nodes' belief states reflect this new information. Many observational vertices will be reached under different belief states so many thousands of new nodes get created. Many of these have a low probability of occurring, but have similar belief states (i.e. are close in the belief space) and often share the same action. We devised an extension to LAO* which takes advantage of this with by weakening the definition of equality to allow two nodes for the same graph vertex to be equal if the belief states are similar. The Kullback-Leibler (KL) divergence is a common measure of the difference between two probability distributions, although not a true distance measure due to its asymmetry. We therefore use an adapted, symmetrised (Seghouane and Amari 2007) KL-divergence to measure the distance between belief states:

$$\mathrm{KL}_{\mathrm{sym}}(b, b') = \frac{1}{2}\sum_i b_i \log_2(\frac{b_i}{b'_i}) + \frac{1}{2}\sum_i b'_i \log_2(\frac{b'_i}{b_i})$$

Two belief states must have a KL divergence lower than a preset maximum, $\max_{\mathrm{KL}}$, to be considered equal. On node expansion, if a node with a belief state within $\max_{\mathrm{KL}}$ already exists an edge is created to it instead of creating a new node.

During simulation of a policy, the belief state is maintained along with the agent's current node in the LAO* graph. When an observation is made the simulator computes the new belief state. The agent's next node is chosen from the current node's successors by selecting the node with the lowest KL divergence from the new belief state.

Approximate Models

Although the MDP approach is faster than the POMDP solver applied to the same problem, the MDPs it produces may be too large to be practically solved. We propose two approaches for reducing the state space and hence the computational requirements at a cost in terms of optimality.

State Space Reduction

For a PRM graph with n vertices, m uncertain edges, and a discretisation into d discrete values, the total number of valid (belief distributions summing to one) MDP states is:

$$n\frac{(d+2^m-2)!}{(d-1)!(2^m-1)!} \tag{4}$$

For a problem with 20 vertices in the graph, four uncertain edges, and a discretisation of only 0.1 (11 values), this results in around 65 million states.

Examining Equation 4, we see that the major contributors to the number of states are the discretisation and the number of worlds. We can reduce the discretisation to make the MDP smaller, but this produces insufficient discretisation to represent the probabilities accurately. A better approach is to reduce the number of continuous variables needed to represent the belief state. The easiest way to do this is to assume that the likelihood of each edge being blocked is independent of all others. With this assumption, we don't need to maintain a belief distribution over all the possible worlds, but instead we keep independent distributions for each edge. This reduces the number of states from that given in Equation 4 to nd^m, so our example above can be represented using under 300 thousand discrete states.

The changes that need to be made to the MDP formulation for this independent model are to redefine the set of discretised belief states, $\widetilde{P}$. Rather than being the set of possible probability distributions over all possible worlds, this is now the set of products of individual distributions for each edge, so $\widetilde{P} = \mathcal{P}(\widetilde{P}_E)$ where $\widetilde{P}_E = \{\widetilde{P}(e_1), \ldots, \widetilde{P}(e_m)\}$ where $\{e_1, \ldots e_m\}$ is the set of uncertain edges. This then changes the definition of the states space S, the reward function R, and the transition function.

For the transition function, the change that needs to be made is in the definition of $\widetilde{P}(w)$ for a world w. This is now defined (by a slight abuse of notation) as:

$$\widetilde{P}(w) = \prod_{e \in w} \widetilde{P}(e) \prod_{e \notin w} (1 - \widetilde{P}(e)) \tag{5}$$

where by $e \in w$ we mean all the uncertain edges that are collision-free in w. The rest of the MDP definition remains the same, with Equation 5 substituted into Equations 1 to 3.

While the independent MDP formulation has a very significant advantage in terms of the number of states and hence the size of problems that can be solved, the policies it produces can be much worse than the dependent MDP formulation. If two edges are perfectly correlated and the agent observed one of them to be blocked, in the dependent case it would know the other was too. In the independent case this correlation can't be represented, so the agent might travel to the second edge on the assumption it is collision free.

Clustering Edges

In the PRM formulation of the problem, the obstacles cause the edges to be blocked. This means that if two edges are very close to one another in the PRM graph and are close to the same obstacle, it is very likely that their probabilities of

being blocked are dependent. On the other hand, two edges that are far apart in the PRM graph are almost certainly independent. This observation leads us a second state reduction technique that lies between the extremes of complete dependence and independence. We want to retain the benefit of representing edge dependencies where they exist, but reduce the complexity of the belief space by not representing dependencies between independent edges. The idea is to cluster together edges that are close to one another, while leaving far away edges independent. Intuitively this makes the space of possible worlds the cross product of a set of smaller clusters of edges.

The clustered formulation allows the agent to infer information about edges it hasn't directly received an observation about via dependence in the belief state, but still to benefit from the computational advantages of ignoring dependencies for independent edges. With clustering, the quality of the policy doesn't suffer to the same extent as with an assumption of total independence between edges.

For brevity, we omit the definition of the clustered MDP here, but intuitively, it is analogous with the independent case above. For each cluster of n uncertain edges, there is a continuous variable for each of the 2^n worlds, but these are independent of the other clusters.

Theorem 1. *The optimal policy for the dependent MDP is no worse than the optimal policy for the clustered and independent MDPs (neglecting the effects of discretisation).*

We prove this by observing that since every state in the independent (or clustered) MDP is also represented in the dependent MDP, and since the action space is the same, if a state in the independent MDP has a better action than the corresponding state in the dependent MDP, that action would also be available in the dependent MDP. For it to be better in the independent MDP, it must have higher value, but then it would have a higher value in the dependent MDP than the optimal action, which is a contradiction.

Corollary 1. *The optimal policy for the clustered MDP is no worse than the optimal policy for the independent MDP.*

Related Work

Approximate POMDP solving To improve the scalability of POMDP planners many approximate solution algorithms have been devised. These algorithms commonly differ from exact solvers by only computing the value function over a subset of belief space. Modern point-based techniques use the starting belief state to explore only those belief points that are reachable by the agent and perform value function updates only at those points. A recent efficient point-based solver is PERSEUS (Spaan and Vlassis 2005). PERSEUS differs from earlier point-based techniques (Pineau, Gordon, and Thrun 2003) by maintaining a fixed set of belief states, B, to plan for, as opposed to interleaving expanding the set of belief points with performing value backups on them. The set B is created by letting the agent perform random walks through the environment from the start state and recording the belief states visited. The value function is then updated by repeatedly selecting a random sub-set of the points in B, computing a new vector over those belief points and adding this vector to the new value function if it is better than the current value function's estimate. As each new vector improves the value estimate for some subset of B, this subset can be ignored for the rest of the current iteration. New value vectors are added until all $b \in B$ have been improved. PERSEUS has been shown to achieve results comparable to other optimal solvers, often in a much quicker time. We use a Matlab implementation of PERSEUS[2] (all other implementations are in Java) in our experiments to compare a point-based POMDP algorithm to the MDP models.

Static Route Planning Substantial research into uncertain motion planning exists, but we restrict ourselves to domains with deterministic actions but unknown obstacles since this has received less attention in the literature. Standard navigation approaches operate directly on the PRM graph and use uncertainty about the obstacle positions to compute a static route offline, so don't account for future information gains. An example is Burns and Brock (2006), who explore map uncertainty by using a lazy approach to roadmap construction which builds roadmaps as queries are evaluated. Roadmap refinement techniques are employed that increase the detail of sensing in tricky areas if a generated route falls below a preset confidence threshold. The idea of associating a "success probability" with edges is used by several implementations and is similar to the first stage of processing used in (Nielsen and Kavraki 2000).

The Minimum Collision Cost (MCC) planner (Missiuro and Roy 2006) plots a static route over a PRM graph (using A*) by considering the cost of a collision on each edge. Instead of minimising total path cost the algorithm seeks to minimise the expected collision cost over the graph. The following equation is used instead of the standard c_e:

$$c_e^{MCC} = P_{col}(e) * C_{const} + (1 - P_{col}(e)) * c_e$$

where $P_{col}(e)$ is the independent prior collision probability of traversing edge e and C_{const} is a fixed cost of collision. Higher values for C_{const} make the MCC planner more conservative about including uncertain edges in its route.

To allow for a fairer comparison in our experiments, we allow the agent to replan using MCC if it reaches a vertex connected to an uncertain edge and finds that edge blocked. Before the MCC planner is invoked, $P_{col}(e)$ for each uncertain edge is set to the same collision probability as in the POMDP start state, ensuring both approaches receive the same initial knowledge.

Experiments

To investigate the effectiveness of the three MDP models and approximate LAO*, we generated six random PRM graphs with 40 vertices and a maximum of four edges per vertex ($\max_{c_e} = 200$) using the standard PRM algorithm. These produce POMDP that are too large to solve exactly to find the optimal policy. Each graph had five uncertain edges, and since all edges were generated to be outside the mean positions of the obstacles, the prior probabilities that

[2]Available from `http://staff.science.uva.nl/~mtjspaan/pomdp/index_en.html`

the uncertain edges were traversable were always at least 0.5. Starting belief states were handcrafted to ensure some edges were dependent on each other while others were independent. Unlike many other applications, random graphs are good indicators of real world performance here because the PRM algorithm works with random graphs. Unfortunately, the randomness (in this case mostly due to the random positioning of obstacles) leads to widely varying performance, making a statistical analysis of performance difficult.

Since vertices that don't provide observations have no effect on the policy we perform a pre-processing stage on each PRM graph to remove these inconsequential vertices (except v_S and v_G) and add edges between observational vertices with c_e equal to the shortest route cost between them. This reduces $|S|$ for the POMDP by $h2^m$ for h removed vertices.

We ran standard and approximate LAO* with the independent, clustered and dependent models as well as the MCC planner and PERSEUS. Each graph had a number of uncertain edges where the probabilities were correlated and the clustered model was constructed so that only some of the correlated edges were clustered while others were treated as independent (otherwise the clustered and dependent MDPs would have produced the same policy). Table 1 compares the algorithms on each graph, showing the average and maximum cost to reach the goal and policy generation times.

We can't compare policies directly by their value functions because these don't reflect the true, dependent model. The independent (and clustered) MDPs compute their policy on the assumption that edges are independent so will obtain different values which may not be correct. All policies were evaluated using 50,000 simulations in the full dependent model. All experiments were run on a 2.33Ghz Xeon E5345 with 500MB of process memory, and with $d = 1 \times 10^{-5}$ and discount factor $\gamma = 0.999$ for LAO*. For the POMDP, a value function tolerance of $\epsilon = 1 \times 10^{-3}$ was used with $\gamma = 0.95$ to ensure convergence in feasible time. The MCC planner treats each edge independently ignoring any dependencies—observing an edge never gives the agent any information about any other edge.

As the results show, there is considerable difference between the graphs. In all cases, the MCC planner is the fastest by a wide margin, but policy performance never surpasses the independent MDP. Since MCC never considers the value of future information it can rarely find an optimal policy. In graphs 1 and 3 it matches the independent MDP, which suggests that the optimal policy under independence is to first try the shortest route to the goal, attempting successively longer routes if blocked edges are observed. When edge dependencies are represented in the alternate MDP models, lower cost policies can be found. MCC also requires selecting a suitable C_{const}, which is highly problem dependent, lacking a simple way to choose 'good' values to guarantee performance. High values make the planner avoid most uncertain edges, choosing safe routes with higher cost.

PERSEUS achieves good results in all but one case, but requires orders of magnitude more computation time and is therefore infeasible for most problems. Only in graphs 1 and 2 does it fail to match the dependent MDP in average cost. Its variance for graph 2 was 0 showing the same route

		LAO*			App.LAO*			MCC	PRS.
		I	C	D	I	C	D		
Grph 1	**Avg**	**1057**	**1018**	**F**	**1055**	**1019**	**1018**	**1054**	**1045**
	Max	2278	1929	F	2278	1929	1929	2278	2301
	T	276	47182	F	76	250	660	< 2	6h41m
Grph 2	**Avg**	**1475**	**1475**	**1470**	**1475**	**1477**	**1470**	**1523**	**1813**
	Max	2097	2097	1993	2097	2097	1993	2124	1813
	T	158	220	2302	182	100	330	< 1	48m
Grph 3	**Avg**	**988**	**913**	**F**	**987**	**911**	**912**	**986**	**911**
	Max	1431	1360	F	1431	1360	1362	1430	1351
	T	544	36656	F	130	234	848	< 3	2h59m
Grph 4	**Avg**	**1212**	**1212**	**1212**	**1211**	**1212**	**1212**	**1217**	**1210**
	Max	1452	1452	1452	1452	1452	1452	1451	1451
	T	92	226	2020	108	130	346	< 1	1h17m
Grph 5	**Avg**	**F**	**F**	**F**	**1603**	**1612**	**1613**	**1650**	**1570**
	Max	F	F	F	1893	3326	3236	1891	1880
	T	F	F	F	1774	812	1784	< 3	7h53m
Grph 6	**Avg**	**1501**	**1426**	**F**	**1499**	**1425**	**1426**	**1526**	**1426**
	Max	2215	2123	F	2215	2123	2123	2062	2123
	T	1422	5686	F	198	300	798	< 2	8h24m

Table 1: A comparison of algorithm performance on six random graphs. I=independent, C=clustered, D=dependent. All times (T) in ms unless stated, averaged over five runs. Average simulation costs taken from 50,000 trials. For approximate LAO*, $\max_{\text{KL}} = 0.1$. MCC $C_{const} = 300$. PERSEUS belief set size is 50,000.

was always followed. In this case it avoided uncertain edges, using a safe, long path. In both cases the poorer performance appears to be due to the point-based approximation.

Clustered and dependent MDP performance matches or exceeds the independent model in all cases. The lower maximum costs attained in graphs 1,3 and 6 indicate the agent was acting better than the independent MDP in trials where some edges were blocked. The observations in these graphs are better exploited by the agent when it can reason about the edge dependencies—less cost is incurred observing uncertain edges correlated to an edge already observed as blocked.

On graphs 2 and 4, the MDP solution qualities were almost identical for all methods (the difference between the independent and clustered costs in approximate LAO* are within the error margin of the simulator). This appears to be because the optimal policies didn't need to use the information about edge dependencies as either the dependent edges weren't used, or the optimal policy visited vertices that gave information about both uncertain edges. In most other examples we see that there is a significant advantage to being able to reason about the correlated edges, but that the reachable state space of the dependent MDP is so much larger that standard LAO* ran out of memory before finding a solution. The geometry of graph 5 is unique in that at least one uncertain edge must be traversed to reach the goal. As our model provides no 'give-up' action, the optimal policy is hard to compute from states where no goal can be reached. This is the most likely reason why standard LAO* fails here.

The dependent model is always closest to optimal, but as our results show, even with five uncertain edges the reachable state space is too large for standard LAO* solve reliably. Approximate LAO* retains the policy quality of standard LAO* while solving significantly quicker. The time advan-

	Graph 1			Graph 6		
$\max_{\mathrm{KL}}$	T(ms)	#Nodes	F	T(ms)	#Nodes	F
S	47182	287853	0	5686	24361	0
0.05	352	1549	0	266	917	0
0.1	250	1022	0	300	918	0
0.2	94	497	5	128	515	0
0.5	62	228	8	84	193	47

Table 2: Comparison of graph sizes for approximate LAO* with varying $\max_{\mathrm{KL}}$. S=Standard LAO*, #Nodes=nodes created, T=time, F=% trials with goal not reached.

tage is clearly highly problem dependent, gaining an order of magnitude speedup in extreme cases, while being slightly slower in two. Observations in the graph strongly affect how much state space must be explored to find the optimal policy and is the main contributor to the variations which can also be seen in the POMDP timings.

Table 2 shows brief results from increasing $\max_{\mathrm{KL}}$on two of the graphs (these two show the most significant effects from changing the approximation threshold). Higher thresholds reduced the number of nodes created, since more distant belief states are treated as equal. Planning time is reduced as fewer node evaluations occur, but this must be traded-off against a reduction in plan quality due to observation information being hidden by approximation, thus hindering the creation of optimal plans. The reduction in plan quality is hard to show directly because as the threshold is increased some policies do not reach the goal in every possible world, leading to infinite policy costs. Instead we show (in the column labelled F in Table 2) the percentage of trials where the policy found does not lead to the goal.

The clustered representation is frequently quite close in terms of performance to the dependent MDP and are generally significantly faster to compute, and approximate representation of the belief space allows for efficient exploration without sacrificing policy quality. The advantages of the clustered representation are that it can be customised to the particular graph—if there are uncertain edges where a significant benefit could be gained by treating them as correlated, they can be clustered while the others can be left independent. It is also more scalable than the dependent MDP to higher numbers of uncertain edges by avoiding doubling the number of possible worlds with each additional edge. One area of future work is to look at how this decision could be made automatically without having to compute the policies.

Conclusions

We have shown that the problem of planning in PRM graphs with uncertain obstacle locations can be thought of as a model identification problem, and can be represented as a POMDP. The POMDP solution isn't viable for real-world problems. PERSEUS, like LAO* gains a computational advantage by rolling out the graph from the starting state to reduce the plan space, however it can't account for the low probability of reaching some states, or for the extra structure in these problems. Although it doesn't suffer from the approximation introduced by discretising the belief state and hence is closer to optimal in some of the graphs, PERSEUS cannot exploit the problem structure as effectively as our approach, so is orders of magnitude slower.

We have shown that this POMDP has a very structured belief space and that it can be represented and solved efficiently using an MDP approximation. We developed several approximations. The dependent MDP produces policies that perform closest to optimal, but the state space grows exponentially as uncertain edges are added, making policy computation harder. A clustered approach retains the advantages of a fully dependent model while scaling to more complex examples by reducing the growth of the state space. When dependencies between edges are weak, the dependent policy doesn't gain much advantage since there is no benefit to representing the additional states. Ignoring small variances in the belief state allows for easier exploration of the belief space to obtain good policies in a much quicker time. An important area for future work will be to test the scalability of approximate LAO* extension to larger graphs with more uncertain edges as these are the largest contributing factor to the state space size.

References

Barto, A. G.; Bradtke, S. J.; and Singh, S. P. 1993. Learning to act using real-time dynamic programming. Technical Report UM-CS-1993-002, University of Massachusetts, Amherst MA 01003.

Burns, B., and Brock, O. 2006. Sampling-based motion planning using uncertain knowledge. Technical report, University of Massachusetts Amherst.

Dean, T.; Kaelbling, L. P.; Kirman, J.; and Nicholson, A. 1995. Planning under time constraints in stochastic domains. *Artificial Intelligence* 76(1–2):35–74.

Hansen, E. A., and Zilberstein, S. 2001. LAO * : A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence* 129(1-2):35–62.

Kavraki, L. E., and Latombe, J. 1998. *Probabilistic roadmaps for robot path planning*. John Wiley, West Sussex, England. 33–53.

Missiuro, P., and Roy, N. 2006. Adapting probabilistic roadmaps to handle uncertain maps. In *Proceedings 2006 IEEE International Conference on Robotics and Automation*, 1261–1267.

Nielsen, C., and Kavraki, L. E. 2000. A two level fuzzy PRM for manipulation planning. In *The IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, 1716–1722. IEEE Press.

Pineau, J.; Gordon, G.; and Thrun, S. 2003. Point-based value iteration: An anytime algorithm for POMDPs. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1025 – 1032.

Ross, S.; Pineau, J.; Paquet, S.; and Chaib-draa, B. 2008. Online planning algorithms for POMDPs. *Journal of Artificial Intelligence* 32:663–704.

Roy, N.; Gordon, G.; and Thrun, S. 2005. Finding approximate pomdp solutions through belief compression. *Journal of Artificial Intelligence Research* 23:1–40.

Seghouane, A. K., and Amari, S. I. 2007. The AIC criterion and symmetrizing the kullback-leibler divergence. *IEEE Transactions on Neural Networks* 18(1):97–106.

Spaan, M. T. J., and Vlassis, N. 2005. Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence* 24:195–220.